Documentation for PDF Text Extraction and Translation Chatbot

This documentation provides an overview of the PDF text extraction and translation chatbot application developed using Streamlit. The application allows users to upload multiple PDF documents, extract text from them, clean the text, translate it, and interact with the extracted information through a conversational interface.

Introduction

The PDF text extraction and translation chatbot application allows users to upload PDF documents, extract text using OCR (Optical Character Recognition), clean the extracted text, translate it, and interact with the translated text through a chat interface. The chatbot leverages LangChain and HuggingFace models to provide a conversational retrieval chain for enhanced user interactions.

Code Overview

Libraries and Modules

- **Streamlit**: For building the web application.
- **dotenv**: For loading environment variables.
- **PyMuPDF (fitz)**: For handling PDF documents.
- **PIL (Pillow)**: For image processing.
- **OpenCV**: For additional image processing.
- **PyTesseract**: For OCR text extraction.
- **NumPy**: For numerical operations.
- **LangChain**: For conversational retrieval chain.
- **HuggingFace Transformers**: For embeddings and language models.
- **googletrans**: For text translation.

Functions

1. **`get_pdf_text(pdf_docs, zoom_x=2.0, zoom_y=2.0)`**:
   - Extracts text from the provided PDF documents using OCR.
   - Parameters: `pdf_docs` (list of PDF files), `zoom_x` (horizontal zoom factor), `zoom_y` (vertical zoom factor).
   - Returns: Extracted text.
2. **`clean_text(text)`**:
   - Cleans the extracted text by removing extra whitespace, special characters, and unwanted elements.
   - Parameters: `text` (string of extracted text).
   - Returns: Cleaned text.

3. **`get_text_chunks(text)`**:
    - Splits the cleaned text into chunks for processing.
    - Parameters: `text` (string of cleaned text).
    - Returns: List of text chunks.
4. **`get_vectorstore(text_chunks)`**:
    - Creates a vector store from the text chunks using embeddings.
    - Parameters: `text_chunks` (list of text chunks).
    - Returns: FAISS vector store.
5. **`get_conversation_chain(vectorstore)`**:
    - Sets up a conversational retrieval chain using the vector store.
    - Parameters: `vectorstore` (FAISS vector store).
    - Returns: Conversational retrieval chain.
6. **`translate_text(text, target_language="en")`**:
    - Translates the provided text into the target language.
    - Parameters: `text` (string of cleaned text), `target_language` (language code for translation, default is English).
    - Returns: Translated text.
7. **`handle_userinput(user_question)`**:
    - Handles user input, processes the question, and generates a response.
    - Parameters: `user_question` (string of user question).
    - No return value (updates session state with the response).

Streamlit App Structure

1. **Main Function (`main`)**:
    - Sets up the Streamlit app configuration.
    - Initializes session state variables.
    - Handles user inputs and displays chat messages.
    - Provides a sidebar for uploading and processing PDF documents.
2. **Sidebar**:
    - Allows users to upload PDF documents.
    - Processes the uploaded documents and extracts, cleans, chunks, and stores the text in a vector store.
    - Sets up the conversational retrieval chain.
3. **Chat Interface**:
    - Displays chat messages from the user and the AI assistant.
    - Allows users to input questions and interact with the extracted and translated text.

Usage

1. **Upload PDFs**:
   - Use the sidebar to upload one or more PDF documents.
   - Click on "Process" to extract and process the text from the uploaded documents.
2. **Ask Questions**:
   - Use the chat input box to ask questions about the uploaded documents.
   - The chatbot will respond with relevant information extracted and translated from the PDFs.

Conclusion

This PDF text extraction and translation chatbot application provides a seamless way to interact with and extract valuable information from PDF documents. By leveraging OCR, text cleaning, translation, and conversational retrieval chains, the application enhances the user's ability to obtain and understand information from their documents.